Remarks

The above Amendments and these Remarks are in reply to the Office Action mailed June 27, 2007. Claims 1-65 are presented herewith for consideration. Applicants have amended claims 1-3, 5-6, 8-10, 12-13, 16-17, 2-21, 23-24, 26, 33, 35-36, 38, 42-43, 45-46, 49, 52, 58, 62-63 and 65, and cancelled claims 18, 32, 37 and 59.

I. Oath/Declaration

The Examiner indicates that the Oath/Declaration filed with this application is defective. Applicants submit an Application Data Sheet with this Response identifying the residence of the fourth inventor.

II. Claim Objections

Claims 1-5, 8, 9, 12-16, 21, 23, 24, 32-37, 43, 45, 46, 52 and 63 are objected to because of certain informalities. Applicants have amended each of the above-identified claims to correct the informalities objected to by the Examiner. Thus, Applicants request that the Examiner remove these claim objections.

III. Rejection Of Claims Under 35 U.S.C. §112

Claims 5, 9, 10, 13-15, 17, 18, 21-24, 26, 29-32, 35-51 and 65 are rejected under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicants have amended the rejected claims to meet the requirements set forth under 35 U.S.C. §112. Thus, Applicants request that the Examiner remove these rejections.

IV. Rejection Of Claims Under 35 U.S.C. §102

Claims 1, 2, 4, 6-8, 11-23, 25, 27-33, 36-39, 41, 42, 44, 45 and 47-62 are rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 7,191,441 issued to Abbott et al. ("*Abbott*").

1. Independent Claim 1

Claim 1 recites:

> "creating a serialized representation of application objects in a runtime environment;
>
> building an optimized object code file using the serialized representation, wherein the optimized object code file replaces the initialization code compiled from the application source code; and
>
> loading the optimized object code file into a new runtime environment."

Abbott does not disclose building an "optimized object code file [that] replaces the initialization code compiled from the application source code" that is subsequently loaded into a new run time environment. In contrast, *Abbott* "provides a set of callable routines and tools that allow a user to store away the state of an initialized Java application." Col. 9, lines 10-13. *Abbott* describes its technology as taking a "snapshot" of an application that can be saves for later use. Col. 9, lines 15-18. *Abbott* then, upon receiving a restore command, reloads the application, component by component, into a Java Virtual Machine. *Abbott*, in col. 18, lines 10-21, discloses that:

> the saved state is loaded by a special "javaload" tool, which loads the stored application into a Java VM, bypassing all the initialisation normally associated with bringing up a Java VM. The reloading of the application is performed in stages. The various components of the Java VM are provided with restore commands to restart themselves using the saved information. These components are launched in parallel with the major structures such as the heap and stacks being recreated (as shown in FIG. 8 below), according to the desired timing between the two (e.g. the CL class loader is launched at the same time as the classes are being reconstructed).

The "various components" *Abbott* refers to includes DLL addresses, heap addresses, thread information, IO channels, and so on. The state of each of these components was recorded at the time the application was saved. To restore the application, *Abbott* reloads the application into a Java VM, component by component. For example, *Abbott* begins the restore process by loading each DLL address recorded in the save file into memory. Col. 18, lines 27-28. Upon loading the DLL addresses, *Abbott* loads each saved heap, thread information, IO channel, etc. into memory. After all the components are restored, the suspended application "is ready to run, and control is passed to execution engine of the Java VM." Col. 18, lines 61-63. Unlike the method recited in claim 1, *Abbott* does not build a new object code ("optimized object code file") replacing the initialization code and subsequently load the new object code in a new runtime environment. As discussed above, *Abbott* simply takes a "snapshot" of the state if the initialized application and recreates the application, component by component. Therefore, Applicants respectfully suggest that the method recited in claim 1 is not anticipated by *Abbott*, and request that the Examiner remove this rejection.

2. Dependent Claims 2, 4, 6-8 and 11-19

Dependent claims 2, 4 6-8 and 11-19 depend directly or indirectly from independent claim 1. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 2, 4 6-8 and 11-19 are allowable for at least the reasons set forth above concerning independent claim 1.

3. Independent Claim 20

For at least the same reasons discussed above with regard to claim 1, the method recited in claim 20 is not anticipated by *Abbott*. Namely, *Abbott* does not disclose a serialized representation that "includes memory relations between application objects and a serialization identifier associated with each application object." Therefore, Applicants respectfully request that the Examiner remove this rejection.

4. Dependent Claims 21-23, 25 and 27-32

Dependent claims 21-23, 25 and 27-32 depend directly or indirectly from independent claim 20. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 21-23, 25 and 27-32 are allowable for at least the reasons set forth above concerning independent claim 20.

5. Independent Claim 33

For at least the same reasons discussed above with regard to claim 1, the method recited in claim 33 is not anticipated by *Abbott*. Namely, *Abbot* does not disclose receiving an object code from an application source server and subsequently loading the object code into a runtime environment. Therefore, Applicants respectfully request that the Examiner remove this rejection.

6. Dependent Claim 36-37

Applicants have cancelled claim 37. Dependent claim 36 depends directly or indirectly from independent claim 33. This dependent claim includes all of the limitations of the independent claim from which it depends. Applicants respectfully assert that dependent claim 36 is allowable for at least the reasons set forth above concerning independent claim 33.

7. Independent Claim 38

For at least the same reasons discussed above with regard to claim 1, the method recited in claim 38 is not anticipated by *Abbott*. Namely, *Abbot* does not disclose building an "optimized object code file" and loading the file into a new runtime environment. Therefore, Applicants respectfully request that the Examiner remove this rejection.

8. Dependent Claims 39, 41, 44-45 and 47-51

Dependent claims 39, 41, 44-45 and 47-51 depend directly or indirectly from independent claim 38. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 39, 41, 44-45 and 47-51 are allowable for at least the reasons set forth above concerning independent claim 38.

9. Independent Claim 52

For at least the same reasons discussed above with regard to claim 1, the method recited in claim 52 is not anticipated by *Abbott*. Namely, *Abbott* does not disclose "building a second object code" that is then loaded into a "second runtime environment." Therefore, Applicants respectfully request that the Examiner remove this rejection.

10. Dependent Claims 53-57

Dependent claims 53-57 depend directly or indirectly from independent claim 52. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 53-57 are allowable for at least the reasons set forth above concerning independent claim 53.

11. Independent Claim 58

For at least the same reasons discussed above with regard to claim 1, the method recited in claim 58 is not anticipated by *Abbott*. Namely, *Abbott* does not disclose "building an optimized object code" that replaces the initialization code compiled from the application

source code. Therefore, Applicants respectfully request that the Examiner remove this rejection.

12. Dependent Claims 59-61

Dependent claims 59-61 depend directly or indirectly from independent claim 58. These dependent claims include all of the limitations of the independent claim from which they depend. Applicants respectfully assert that dependent claims 59-61 are allowable for at least the reasons set forth above concerning independent claim 58.

13. Independent Claim 62

For at least the same reasons discussed above regarding claim 1, Applicants respectfully suggest that the method recited in claim 62 is not anticipated by *Abbott*. Namely, *Abbott* does not disclose "building an optimized object code" that replaces the initialization code compiled from the application source code. Therefore, Applicants request that the Examiner remove this rejection.

V. Rejection Of Claims Under 35 U.S.C. §103

Claims 3, 40 and 63-65 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of U.S. Publication No. 2004/0044989 ("*Vachuska*").

Claims 5, 9, 10, 24, 26, 35, 43, 46 and 65 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott*.

Claim 34 is rejected under 35 U.S.C. §103(a) as being unpatentable over *Abbott* in view of U.S. Patent No. 5,987,256 ("*Wu*").

Abbott* in view of *Vachuska

1. Dependent Claim 3

Claim 3 depends from independent claim 1. Claim 1, in part, recites building an "optimized object code file [that] replaces the initialization code compiled from the application source code" that is subsequently loaded into a new run time environment. As discussed above, *Abbott* does not disclose this claim element. *Vachuscka* does not provide the elements missing from *Abbott*. The Examiner cites *Vachuska* to teach obtaining reflective information about classes and objects. However, the technology disclosed in *Vachuska* cannot be incorporated into *Abbott* to disclose "optimized object code file [that] replaces the initialization code compiled from the application source code." *Vachuska* discloses a source-code generator which uses a reflection mechanism to analyze source templates. Nothing disclosed in *Vachuska* can be incorporated into *Abbott* to disclose the method recited in claim 1. Thus, Applicants respectfully suggest that the method recited in claim 1 is not obvious over *Abbott* in view of *Vachuska*. Because claim 3 depends from claim 1, claim 3 is also not obvious over *Abbott* in view of *Vachuska*.

2. Dependent Claim 40

Claim 40 depends from independent claim 38. Claim 38, in part, recites building an "optimized object code file [that] replaces the initialization code compiled from the application source code." For at least the same reasons discussed above regarding claim 3, Applicants respectfully suggest that the method recited in claim 38 is not obvious over *Abbott* in view of *Vachuska*. Because claim 40 depends from claim 38, claim 40 is also not obvious over *Abbott* in view of *Vachuska*.

3. Dependent Claims 63-65

Claims 63-65 depend from independent claim 62. Claim 62, in part, recites building an "optimized object code file [that] replaces the initialization code compiled from the application source code." For at least the same reasons discussed above regarding claim 3,

Applicants respectfully suggest that the method recited in claim 62 is not obvious over *Abbott* in view of *Vachuska*. Because claims 63-65 depend from claim 62, claims 63-65 are also not obvious over *Abbott* in view of *Vachuska*.

Abbott

1. **Dependent Claims 5 and 9-10**

Claims 5 and 9-10 depend directly or indirectly from independent claim 1. Claim 1, in part, recites building an "optimized object code file [that] replaces the initialization code compiled from the application source code" that is subsequently loaded into a new run time environment. As discussed above, *Abbott* does not disclose this claim element.

Moreover, one skilled in the art could not modify *Abbott* to build an "optimized object code file [that] replaces the initialization code compiled from the application source code" that is subsequently loaded into a new run time environment. Therefore, Applicants respectfully suggest that the method recited in claim 1 is not obvious over *Abbott*. Because claims 5 and 9-10 depend from claim 1, claims 5 and 9-10 are also not obvious over *Abbott*.

2. **Dependent Claims 24 and 26**

Claims 24 and 26 depend from independent claim 20. Claim 20, in part recites building an "optimized object code file [that] replaces the initialization code compiled from the application source code" that is subsequently loaded into a new run time environment. For at least the same reasons discussed above regarding claim 1, the method recited in claim 20 is not obvious over *Abbott*. Because claims 24 and 26 depend from claim 20, claims 24 and 26 are also not obvious over *Abbott*.

3. **Dependent Claim 35**

Claim 35 depends from independent claim 33. Claim 33, in part, recites receiving object code and subsequently "loading the object code received from the application source server into a runtime environment." The object code includes "instructions for replicating a runtime memory state of the application." As described above with regard to claim 1, *Abbott* does not create object code replicating the runtime memory state of the application and load the object code into a new runtime environment. In contrast, *Abbott* saves a "snapshot" of the application in the saved state and rebuilds the application component by component when a restore routine is called. Therefore, Applicants respectfully suggest that the method recited in claim 33 is not obvious over *Abbott*. Because claim 35 depends from claim 33, claim 35 is also not obvious over *Abbott*.

4. **Dependent Claim 43 and 46**

Claims 43 and 46 depend from independent claim 38. Claim 38, in part recites building an "optimized object code file [that] replaces the initialization code compiled from the application source code" that is subsequently loaded into a new run time environment. For at least the same reasons discussed above regarding claim 1, the method recited in claim 38 is not obvious over *Abbott*. Because claims 43 and 46 depend from claim 38, claims 43 and 46 are also not obvious over *Abbott*.

5. **Dependent Claim 65**

Claim 65 depends from independent claim 62. Claim 62, in part recites building an "optimized object code file [that] replaces the initialization code compiled from the application source code" that is subsequently loaded into a new run time environment. For at least the same reasons discussed above regarding claim 1, the method recited in claim 62 is not obvious over *Abbott*. Because claim 65 depends from claim 62, claim 65 is also not obvious over *Abbott*.

Abbott* in view of *Wu

Claim 34 depends from independent claim 33. Claim 33, in part, recites receiving object code and subsequently "loading the object code received from the application source server into a runtime environment." The object code includes "instructions for replicating a runtime memory state of the application." As described above with regard to claim 35, *Abbott* does not create object code replicating the runtime memory state of the application and load the object code into a new runtime environment. In contrast, *Abbott* saves a "snapshot" of the application in the saved state and rebuilds the application component by component when a restore routine is called. Therefore, Applicants respectfully suggest that the method recited in claim 33 is not obvious over *Abbott*.

Moreover, *Wu* does not disclose the elements missing from *Abbott*. The Examiner cites *Wu* to teach that object code may include media assets. Applicants respectfully suggest that that technology disclosed in *Wu* cannot be incorporated into *Abbott* to teach the method recited in claim 33. Incorporating object code that includes media assets into *Abbott* does not disclose receiving object code and subsequently "loading the object code received from the application source server into a runtime environment." Therefore, Applicants respectfully suggest that the method recited in claim 33 is not obvious over *Abbott* in view of *Wu*. Because claim 35 depends from claim 33, claim 35 is also not obvious over *Abbott* in view of *Wu*.

Additional Remarks

Based on the above amendments and these remarks, reconsideration of claims ** is respectfully requested.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned attorney by telephone.

Enclosed is a PETITION FOR EXTENSION OF TIME UNDER 37 C.F.R. § 1.136 for extending the time to respond up to and including today, December 26, 2007.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: December 26, 2007

By: */Scott D. Sanford/*
Scott D. Sanford
Reg. No. 51,170

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105
Telephone: (415) 369-9660
Facsimile: (415) 369-9665